Lab 7 – The Model, View, Controller (MVC) Design Pattern and Exceptions

*Computer Science 2334*

***Due by: Friday, 07 April 2017, 11:59 pm***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Members: |  | | | |
|  |  | | | |
|  |  | | | |
|  |  | | | |
|  |  | | | |

# Note on Team Work and Lab Submission:

You ***may*** work on this lab in a team with one or more classmates. You are encouraged (but not required) to work on this lab with your Project 4 teammates. For this lab, you will edit this file in OpenDocument Text format as well as editing the Java project in Eclipse. You should ensure that all team members include their names above and submit it along with the completed Eclipse archive (zip) file in D2L. There should be ***one*** submission for each group.

# Objectives:

1. To learn how to create a model in the MVC design pattern.
2. To learn how to create a view in the MVC design pattern.
3. To learn how to create a controller in the MVC design pattern.
4. To learn how the model, view, and controller are connected to one another and communicate with one another in the MVC design pattern.
5. To learn how to handle exceptions using try … catch blocks.
6. To demonstrate this knowledge by completing a series of exercises.

# Instructions:

This lab exercise requires a laptop with an Internet connection. Once you have completed the exercises in this document, your team will submit it for grading.

Make sure you read this lab description and look at all of the source code posted on the class website for this lab exercise before you begin working.

# Assignment:

Graphical User Interfaces (GUIs) using the Model, View, Controller (MVC) design pattern are an important programming abstraction that shows how additional structure can be built from objects; it is also one that will be used in Projects 4 and 5. Carefully inspect how it works and the documentation comments included in the code.

1. Download the Lab6-eclipse.zip project archive from the class website. Import the project into your Eclipse workspace using the slides from Lab 2. You will submit the modified project archive along with this lab handout modified to include your answers when you are finished.
2. Review the source code posted on the class website. Pay close attention to the MVC classes, which are the Model, StateView, PlaceView, and Controller classes.
3. Read through the source code of the Model class and note the comments provided in the source code that give hints as to what needs to be done in the program.
4. Declare variables called statePlaceList and selectedPlaceList to hold the information in the model for the application objects being modeled. These variables must be of appropriate types to hold lists of Place information. (See the Place class for details on what information is available for each place.) What data structures will be used for statePlaceList and selectedPlaceList and why?

**A List of type Place will be used for both statePlaceList and selectedPlaceList, so we can switch between different types of lists when accessing or utilizing the “list”. List will not allow for duplicate places to be added to the selected places list.**

1. Declare a variable called actionListenerList to hold the information in the model that allows it to be the source for action events. This variable must be of an appropriate type to hold a list of listeners. What data structure will be used for actionListenerList and why?

**An ArrayList of type ActionListener, so we can keep track of the order in which the actions are performed.**

1. Complete the Model class by adding a call to its processEvent() method. This method should be called with an ActionEvent that has "Addition to Selected Places" as its command string. Where should processEvent() be called? Why?

**processEvent() should be called in addToSelectedList method to notify the view that a new place was added to the selected place list.**

What arguments should be provided to processEvent()? Why?

**A new actionEvent of action performed because the action of adding to a list was performed by the addToSelectedList method.**

1. Read through the source code of the StateView class and note the comments provided in the source code that give hints as to what needs to be done in the program.
2. Complete the setModel() method for StateView by registering the view.

Will the StateView object be a source or a listener in this part of the code? Why?

**The StateView object will be a listener, so it can listen for changes in data from the model.**

To which class will the object at the other end of this source/listener connection belong? Why? Will

that other object be a source or a listener? Why?

**It will connect to a source in the class Model, because a source tells a listener to create an event depending on encapsulated data changes in the Model.**

1. Read through the source code of the PlaceView class and note the comments provided in the source code that give hints as to what needs to be done in the program.
2. Complete the setModel() method for PlaceView by registering the view. Will the PlaceView object be a source or a listener in this part of the code? Why?

**The PlaceView object will be a listener, so it can listen for changes in data from the model.**

To which class will the object at the other end of this source/listener connection belong? Why? Will

that other object be a source or a listener? Why?

**It will connect to a source in the class Model, because a source tells a listener to create an event depending on encapsulated data changes in the Model.**

1. Read through the source code of the Controller class and note the comments provided in the source code that give hints as to what needs to be done in the program.
2. Complete the setView() method for the controller by registering an object of the inner class in the controller. Will this object from the Controller be a source or a listener in this part of the code? Why?

**It will be a listener, which will listen to the select place button.**

To which class will the object at the other end of this source/listener connection belong? Why? Will

that other object be a source or a listener? Why?

**It will be a source from the StateView class, because listeners listens to sources. The source for the button listener is the select state button in StateView.**

1. Handle exceptions appropriately using try … catch blocks. Find all methods that throw exceptions (FileNotFoundException, IOException, etc.). List those methods here:

**Model class: readFile**

**Lab6Driver: Lab6Driver, main**

1. Choose where in your code to handle each exception. Explain your design choice here:

**We chose to print the stack trace, so the programmer can see where the exceptions came from.**

1. Handle each exception using an appropriate try … catch block. Remove any unnecessary throws statements. Explain which throws statements you removed and why:

* **Removed IOException from Lab6Driver, because there is no input or output of data to different classes.**
* **Removed both IOException and FileNotFoundException in main, because any errors in main will be caught in Lab6Driver constructor.**
* **Removed IOException in readFile method of Model, because there is no input or output from readFile.**

1. Test the program to make sure it correctly responds to list selection events.
2. Ensure that there are no warnings generated for your code. **Do not suppress warnings.** Fix your code so that warnings are not necessary. (If you can't figure out how to fix your code to avoid the cast warning on the cloned actionListenerList, you may leave in that warning.)
3. Submit the **project archive** along with your completed copy of this lab handout following the steps given in the **Submission Instructions** by **11:59 pm on Friday, 07 April 2017** through D2L ([http://learn.ou.edu](http://learn.ou.edu/)).